IN THE CLAIMS

For the convenience of the Examiner, all pending claims of the present Application are presented below whether or not an amendment has been made. Please amend the claims as follows:

1. **(Currently Amended)** A method of simulating system conditions at a kernel-level, comprising:

intercepting an operating system call from ~~an application~~ **a software application** at a kernel-level;

determining ~~whether the~~ **that a first** operating system call was called from a process that was identified for failure emulation **for the testing of the software application**;

~~if the operating system call was called from process that was identified for failure emulation,~~ consulting user loaded rules and returning results to the **first** operating system call according to the user loaded rules **to result in the testing of the software application**; and

~~if the operating system call was not called from a process that was identified for failure emulation,~~ calling a native operating system service routine associated with ~~the~~ **a second** operating system call**, the testing of the software application in response to the first operating system call not effecting the calling of the native operating system service routine associated with the second operating system call**.

2. **(Original)** The method of claim 1, wherein the determining includes at least determining whether an identifier of the process was communicated previously as being a process for failure emulation.

3. **(Currently Amended)** The method of claim 1, wherein the user loaded rules include **at least one rule directed to the group consisting of** type of failure to emulate, frequency of failure, ~~or~~ **and** error codes to return~~, or combination thereof~~.

4. **(Original)** The method of claim 1, wherein the intercepting is transparent to the process for failure emulation.

5. **(Currently Amended)** A method of simulating system conditions at a kernel-level, comprising:

identifying one or more processes to a kernel-level module for which to emulate failures **for the testing of one or more software applications**;

transmitting one or more failure rules to the kernel-level module, the one or more failure rules associated with the one or more processes;

activating the kernel-level module; and

running the one or more processes **to result in the testing of the one or more software applications, the running of the one or more processes not effecting the overlapping performance of one or more native operating system service routines**.

6. **(Original)** The method of claim 5, wherein the identifying includes at least transmitting one or more process identifiers to the kernel-level module, the one or more process identifiers associated with respective one or more processes for which failure emulation is to be performed.

7. **(Original)** The method of claim 5, further including deactivating the kernel-level module.

8. **(Currently Amended)** ~~A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method of simulating system conditions at a kernel-level, comprising~~ **Logic for simulating system conditions at a kernel level, the logic encoded in a storage medium and operable when executed to**:

~~intercepting~~ **intercept** an operating system call from ~~an application~~ **a software application** at a kernel-level;

~~determining whether the~~ **determine that a first** operating system call was called from process that was identified for failure emulation **for the testing of the software application**;

~~if the operating system call was called from a process that was identified for failure emulation, consulting~~ **consult** user loaded rules and returning results to the **first** operating system call according to the user loaded rules **to result in the testing of the software application**; and

~~if the operating system call was not called from a process that was identified for failure emulation, calling~~ **call** a native operating system service routine associated with ~~the~~ **a second** operating system call**, the testing of the software application in response to the first operating system call not effecting the calling of the native operating system service routine associated with the second operating system call**.

9. **(Currently Amended)** The ~~program storage device~~ **logic** of claim 8, wherein the determining includes at least determining whether an identifier of the process was communicated previously as being a process for failure emulation.

10. **(Currently Amended)** The ~~program storage device~~ **logic** of claim 8, wherein the user loaded rules include **at least one rule directed to the group consisting of** type of failure to emulate, frequency of failure, ~~or~~ **and** error codes to return~~, or combination thereof~~.

11. **(Currently Amended)** The ~~**program storage device**~~ **logic** of claim 8, wherein the intercepting is transparent to the process for failure emulation.

12. **(Currently Amended)** A system for simulating system conditions at a kernel-level, comprising:

a user-space module operable to transmit one or more process identifiers and one or more rules associated with the process identifiers for emulating failure conditions **for the testing of a software application** at a kernel-level; and

a kernel-level module ~~**operable to**~~ **operable to:**

intercept **an operating** system call **from the software application**, ~~**and further operable to**~~ determine ~~**whether the**~~ **that a first operating** system call was invoked from one or more processes identified by the one or more process ~~**identifiers**~~ **identifiers,**

~~**and if the system call was invoked from the one or more processes identified by the one or more process identifiers, the kernel-level module further operable to**~~ **consult the one or more rules associated with the process identifiers and** generate a return result according to the one or more rules **to result in the testing of the software application**, and

~~**if the system call was not invoked from the one or more processes identified by the one or more process identifiers, the kernel-level module further operable to**~~ call ~~**native**~~ **a native** operating system service routine associated with ~~**the**~~ **a second operating** system call**, the testing of the software application in response to the first operating system call not effecting the calling of the native operating system service routine associated with the second operating system call**.

13. **(Original)** The system of claim 12, wherein the user-space module further includes an application programming interface that communicates with the kernel-level module.

14. **(New)** The method of claim 1, wherein intercepting the operating system call comprises replacing an address of operating system call with an address of a process associated with the user loaded rules.

15. **(New)** The method of claim 5, wherein the one or more failure rules include at least one rule directed to the group consisting of type of failure to emulate, frequency of failure, and error codes to return.

16. **(New)** The method of claim 5, wherein the running of the one or more processes is transparent to the process for failure emulation.

17. **(New)** The method of claim 5, wherein running of the one or more processes comprises:

intercepting an operating system call from the one or more software application at a kernel-level; and

replacing an address of the operating system call with an address of a process associated with the one or more failure rules.

18. **(New)** The system of claim 12, wherein the one or more rules include at least one rule directed to the group consisting of type of failure to emulate, frequency of failure, and error codes to return.

19. **(New)** The system of claim 12, wherein the intercepting of the operating system call is transparent to the process for failure emulation.

20. **(New)** The system of claim 12, wherein intercepting the operating system call from the one or more software application at a kernel-level comprises replacing an address of the operating system call with an address of a process associated with the one or more rules.